Network Working Group
Internet-Draft
Intended status: Experimental
Expires: August 21, 2015

M. Kerwin

February 17, 2015

HTTP/2 Encoded Data
draft-kerwin-http2-encoded-data-04

Abstract

This document introduces new frame types for transporting encoded data between peers in the Hypertext Transfer Protocol version 2 (HTTP/2), and an associated error code for handling invalid encoding.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 21, 2015.

Table of Contents

1. Introduction

This document introduces a mechanism for applying encoding, particularly compression, to data transported between two endpoints in version 2 of the Hypertext Transfer Protocol (HTTP/2), analogous to Transfer-Encoding in HTTP/1.1 [RFC7230].

1.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Additions to HTTP/2

This document introduces two new HTTP/2 frame types ([I-D.ietf-httpbis-http2], Section 11.2) and a new HTTP/2 error code ([I-D.ietf-httpbis-http2], Section 7), to allow the application of encoding, particularly compression, to data.

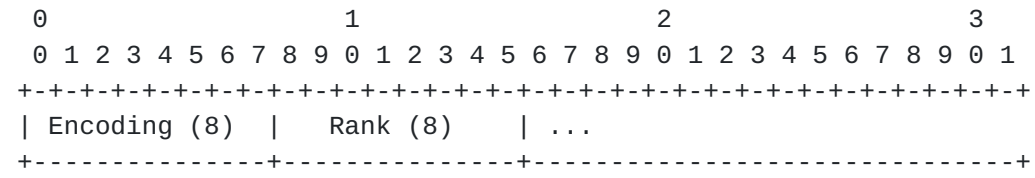
Note that while encoding some or all data in a stream might affect the total length of the corresponding HTTP message body, the "content-length" header, if present, should continue to reflect the total length of the _unencoded_ data. This is particularly relevant when detecting malformed messages ([I-D.ietf-httpbis-http2], Section 8.1.2.5).

2.1. ACCEPT_ENCODED_DATA

An ACCEPT_ENCODED_DATA frame (type code=0xTBA) is used to indicate the sender's ability and willingness to receive ENCODED_DATA frames that are encoded using the schemes identified in its payload.

ACCEPT_ENCODED_DATA always apply to a connection, never a single stream. The stream identifier for an ACCEPT_ENCODED_DATA frame MUST be zero (0x0). If an endpoint receives an ACCEPT_ENCODED_DATA frame whose stream identifier field is anything other than 0x0, the endpoint MUST respond with a connection error (([I-D.ietf-httpbis-http2], Section 5.4.1) of type PROTOCOL_ERROR.

The payload length of an ACCEPT_ENCODED_DATA frame MUST be an exact multiple of 16 bits (2 bytes). An endpoint that receives an ACCEPT_ENCODED_DATA frame with an odd length MUST treat this as a connection error ([I-D.ietf-httpbis-http2], Section 5.4.1) of type PROTOCOL_ERROR.



ACCEPT_ENCODED_DATA Frame Payload

The ACCEPT_ENCODED_DATA frame contains zero or more tuples comprising the following fields:

- Encoding: An 8-bit identifier which identifies the encoding being advertised (see Section 3).

- Rank: An 8-bit integer value.

The rank fulfils the same role as in the HTTP/1.1 TE header ([RFC7230], Section 4.3). The rank value is an integer in the range 0 through 255, where 1 is the least preferred and 255 is the most preferred; a value of 0 means "not acceptable".

The ENCODING_IDENTITY encoding (Section 3) is always acceptable with a default rank of 1, and MUST NOT be advertised with a rank of 0. And endpoint that receives an ACCEPT_ENCODED_DATA frame including an {encoding,rank} tuple of {ENCODING_IDENTITY,0} MUST respond with a connection error ([I-D.ietf-httpbis-http2], Section 5.4.1) of type PROTOCOL_ERROR.

An endpoint that receives an ACCEPT_ENCODED_DATA frame containing an {encoding,rank} tuple with an unknown or unsupported encoding identifier MUST ignore that tuple.

Each ACCEPT_ENCODED_DATA frame fully replaces the set of tuples sent in a previous frame; if an encoding identifier is omitted from a subsequent ACCEPT_ENCODED_DATA frame it is deemed "not acceptable", with the exception of ENCODING_IDENTITY which defaults to a rank of 1.

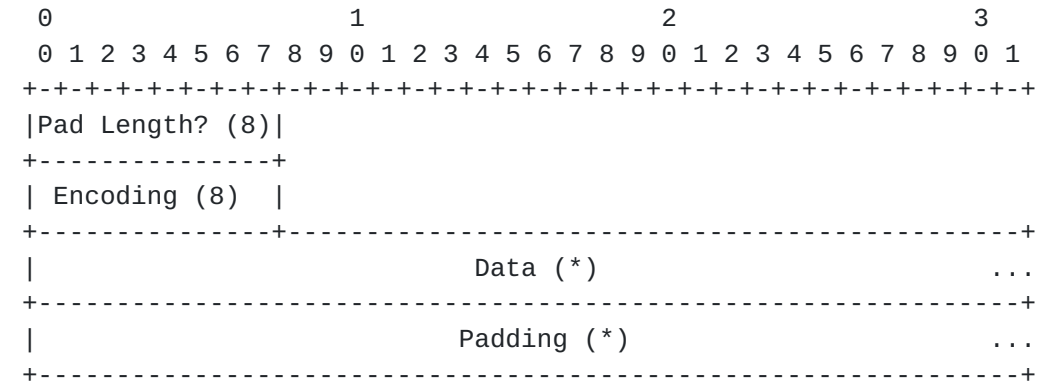
An endpoint may advertise support for an encoding scheme and later decide that it no longer supports that scheme. After sending an ACCEPT_ENCODED_DATA that omits the encoding identifier in question, or includes it with a rank of 0, the endpoint SHOULD continue to accept ENCODED_DATA frames using that scheme for a reasonable amount of time to account for encoded frames that are already in flight.

The ACCEPT_ENCODED_DATA frame does not define any flags, and is not subject to flow control.

2.2. ENCODED_DATA

ENCODED_DATA frames (type code=0xTBA) are semantically identical to DATA frames ([I-D.ietf-httpbis-http2], Section 6.1), but have an encoding applied to their payload. Significantly, ENCODED_DATA frames are subject to flow control ([I-D.ietf-httpbis-http2], Section 5.2).

Any encoding or decoding context for an ENCODED_DATA frame is unique to that frame.



ENCODED_DATA Frame Payload

The ENCODED_DATA frame contains the following fields:

- Pad Length: An 8-bit field containing the length of the frame padding in units of octets. This field is optional and is only present if the PADDED flag is set.

- Encoding: An 8-bit identifier which identifies the encoding that has been applied to the Data field (see Section 3).

- Data: Encoded application data. The amount of encoded data is the remainder of the frame payload after subtracting the length of the other fields that are present.

- Padding: Padding octets that contain no application semantic value. Padding octets MUST be set to zero when sending and ignored when receiving.

The ENCODED_DATA frame defines the following flags:

- "END_STREAM" (0x1): Bit 1 being set indicates that this frame is the last that the endpoint will send for the identified stream. Setting this flag causes the stream to enter one of the "half closed" states or the "closed" state ([I-D.ietf-httpbis-http2], Section 5.1).

- "PADDED" (0x8): Bit 4 being set indicates that the Pad Length field is present.

- "SEGMENT" (0x10): Bit 5 being set indicates that the current segment begins and ends with the current frame.

- "SEGMENT_CONTINUES" (0x20): Bit 6 being set indicates that the current segment continues after the current frame ([I-D.kerwin-http2-segments], Section 2). If the preceding frame did not have the SEGMENT_CONTINUES flag, the current segment begins at the start of the current frame. Intermediaries MUST NOT coalesce frames across a segment boundary and MUST preserve segment boundaries when forwarding frames.

The SEGMENT and SEGMENT_CONTINUES flag MUST NOT be set on any frames unless the remote endpoint has indicated support by sending a SETTINGS_USE_SEGMENTS setting ([I-D.kerwin-http2-segments], Section 3) with a value of 1.

An ENCODED_DATA frame MUST NOT be sent on a connection before receiving an ACCEPT_ENCODED_DATA frame. A sender MUST NOT apply an encoding that has not first been advertised by the peer in an ACCEPT_ENCODED_DATA frame, or was advertised with a rank of 0. Endpoints that receive a frame with an encoding they do not recognise

or support MUST treat this as a connection error of type PROTOCOL_ERROR.

An intermediary, on receiving an ENCODED_DATA frame, MAY decode the data and forward it to its downstream peer in one or more DATA frames. If the received ENCODED_DATA frame has an encoding the downstream peer does not support (as advertised in an ACCEPT_ENCODED_DATA frame) the intermediary MUST decode the data before forwarding it. The intermediary MAY re-encode the data with a scheme supported by the downstream peer and forward it in one or more ENCODED_DATA frames.

If an endpoint detects that the payload of an ENCODED_DATA frame is not encoded correctly according to its declared encoding, for example with a mismatched checksum, the endpoint MUST treat this as a stream error (see [I-D.ietf-httpbis-http2], Section 5.4.2) of type DATA_ENCODING_ERROR (Section 2.3). The endpoint MAY then choose to immediately send an ACCEPT_ENCODED_DATA frame that excludes the encoding in question.

If an intermediary propagates an ENCODED_DATA frame from the source peer to the destination peer without modifying the payload or its encoding, and receives a DATA_ENCODING_ERROR from the receiving peer, it SHOULD pass the error on to the source peer.

ENCODED_DATA frames MUST be associated with a stream. If an ENCODED_DATA frame is received whose stream identifier field is 0x0, the recipient MUST respond with a connection error ([I-D.ietf-httpbis-http2], Section 5.4.1) of type PROTOCOL_ERROR.

ENCODED_DATA frames are subject to flow control and can only be sent when a stream is in the "open" or "half closed (remote)" states. The entire ENCODED_DATA frame payload is included in flow control, including the encoding identifier, and Pad Length and Padding fields if present. If an ENCODED_DATA frame is received whose stream is not in "open" or "half closed (local)" state, the recipient MUST respond with a stream error ([I-D.ietf-httpbis-http2], Section 5.4.2) of type STREAM_CLOSED.

The total number of padding octets is determined by the value of the Pad Length field. If the length of the padding is greater than the length of the remainder of the frame payload, the recipient MUST treat this as a connection error ([I-D.ietf-httpbis-http2], Section 5.4.1) of type PROTOCOL_ERROR.

Note: A frame can be increased in size by one octet by including a Pad Length field with a value of zero.

Padding is a security feature; see Section 10.7 of [I-D.ietf-httpbis-http2].

2.2.1. Fragmentation and Segments

Traversing a network segment with small frame size limits introduces the risk of fragmenting an encoded stream. This can be avoided using segments, as defined in [I-D.kerwin-http2-segments].

An intermediary MAY coalesce multiple adjacent ENCODED_DATA frames if all of them, with the optional exception of the final frame in the sequence, have the SEGMENT_CONTINUES flag set. The coalesced payload MAY be subsequently emitted in any combination of ENCODED_DATA and DATA frames. The payloads of any resulting ENCODED_DATA frames MUST be correctly encoded according to those frames' encodings; note that for some encoding schemes this could require that the payloads of the original frames be decoded and subsequently re-encoded into the new frames, rather than simply concatenated.

2.3. DATA_ENCODING_ERROR

The following new error code is defined:

- "DATA_ENCODING_ERROR" (0xTBA): The endpoint detected that its peer sent an ENCODED_DATA frame with an invalid encoding.

3. Encoding Schemes

The following encoding schemes are defined:

- "ENCODING_IDENTITY" (0): The identity encoding identifier is used to indicate that no encoding is applied.

- "ENCODING_GZIP" (1): The gzip coding is an LZ77 coding with a 32 bit CRC that is commonly produced by the gzip file compression program [RFC1952].

4. Security Considerations

Further to the Use of Compression in HTTP/2 ([I-D.ietf-httpbis-http2], Section 10.6), intermediaries MUST NOT apply compression to DATA frames, or alter the compression of ENCODED_DATA frames other than decompressing, unless additional information is available that allows the intermediary to identify the source of data. In particular, frames that are not compressed cannot be compressed, and frames that are separately compressed can only be merged into a single compressed frame if they occupy the same segment.

5. IANA Considerations

This document updates the registries for frame types and error codes in the "Hypertext Transfer Protocol (HTTP) 2 Parameters" section. This document also establishes a new registry for HTTP/2 encoding scheme codes. This new registry is entered into the "Hypertext Transfer Protocol (HTTP) 2 Parameters" section.

5.1. HTTP/2 Frame Type Registry Update

This document updates the "HTTP/2 Frame Type" registry ([I-D.ietf-httpbis-http2], Section 11.2). The entries in the following table are registered by this document.

Frame Type	Code	Section
ACCEPT_ENCODED_DATA	TBD	Section 2.1
ENCODED_DATA	TBD	Section 2.2

5.2. HTTP/2 Error Code Registry Update

This document updates the "HTTP/2 Error Code" registry ([I-D.ietf-httpbis-http2], Section 11.4). The entries in the following table are registered by this document.

Name	Code	Description	Specification
DATA_ENCODING_ERROR	TBD	Invalid encoding detected	Section 2.3

5.3. HTTP/2 Encoding Schemes Registry

This document establishes a registry for encoding scheme codes. The "HTTP/2 Encoding Schemes" registry manages an 8-bit space. The "HTTP/2 Encoding Schemes" registry operates under either of the "IETF Review" or "IESG Approval" policies [RFC5226] for values between 0x00 and 0xef, with values between 0xf0 and 0xff being reserved for experimental use.

New entries in this registry require the following information:

- Frame Type: A name or label for the encoding scheme.

- Code: The 8-bit code assigned to the encoding scheme.

- Specification: A reference to a specification that includes a description of the encoding scheme.

An initial set of encoding scheme code registrations can be found in Section 3.

6. Acknowledgements

Thanks to Keith Morgan for his advice, input, and editorial contributions.

7. References

7.1. Normative References

[I-D.ietf-httpbis-http2]
Belshe, M., Peon, R., and M. Thomson, "Hypertext Transfer Protocol version 2", draft-ietf-httpbis-http2-17 (work in progress), February 2015.

[I-D.kerwin-http2-segments]
Kerwin, M., "HTTP/2 Segments", draft-kerwin-http2-segments-02 (work in progress), February 2015.

[RFC1952] Deutsch, P., Gailly, J-L., Adler, M., Deutsch, L., and G. Randers-Pehrson, "GZIP file format specification version 4.3", RFC 1952, May 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

7.2. Informative References

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

Author's Address

Matthew Kerwin

Email: matthew@kerwin.net.au
URI: http://matthew.kerwin.net.au/